High-Throughput LDPC Decoders Using A Multiple Split-Row Method

T. Mohsenin and B. M. Baas

ECE Department, University of California, Davis

Abstract— ***We propose the "Multi-Split-Row" LDPC decoding method which allows further reductions in routing complexity, greater throughput, and smaller circuit area implementations compared to the previously proposed Split-Row decoding method. Multi-Split-Row is especially useful for regular high row weight LDPC codes. A 2048-bit full parallel decoder is implemented in a 0.18 μm CMOS technology using standard MinSum, Split-Row-2 and Split-Row-4 methods. The Split-Row-4 decoder delivers 7.1 Gbps throughput with 15 decoding iterations, and has 3.2 times smaller circuit area and 5.2 times higher throughput than the standard MinSum decoder.***

Keywords: LDPC codes, Digital signal processors, Parallel architectures, Parallel algorithms, Very-large-scale integration

I. Introduction

Low density parity check codes first introduced by Gallager [1] have recently received significant attention due to their significant error correction performance and their inherent parallelizable decoder architectures. Many recent communication standards such as 10 Gigabit Ethernet (10GBASE-T) [2] and digital video broadcasting (DVB-S2) [3] have adopted LDPC codes.

Full-parallel decoders which directly map row and column processors together through their Tanner graph interconnection network provide high throughputs [4], [5], [6], [7]. However, due to a large number of processing units and wires between them, they suffer from a large hardware and interconnect complexity. In partially-parallel decoders, multiple row and column processors share one processor and one memory unit. Block-structured LDPC codes [8], [9] and scheduling algorithms [10], [11] proposed for these architectures provide overlapped and reordered row and column processing stages to reduce the processing time which results in decoding throughputs in the range of tens to hundreds of Mbps. While these architectures provide a range of trade-offs between hardware complexity and throughput, they are not well suited to high throughput applications.

The recently proposed *Split-Row* decoding method [7] for regular LDPC codes decreases interconnect complexity by splitting the rows of the parity check matrix into nearly independent halves and provides a higher throughput and smaller interconnect complexity.

This paper introduces the *Multi-Split-Row* method to further enhance the throughput and energy efficiency, and is organized as follows: Section 2 reviews LDPC codes and the message passing algorithm. Section 3 proposes multiple splitting method for regular high-row weight LDPC codes and shows the error performance comparison for different codes with the multiple splitting method. The mapping architecture of the multiple splitting method is presented in Section 4. In Section 5 the results of a full parallel decoder implemented with the proposed and standard techniques are presented and compared.

II. LDPC Codes and Message Passing Decoding

LDPC codes are defined by an $M \times N$ binary matrix called the parity check matrix H, where M defines the number of parity check equations for the code and N defines the code length. LDPC codes are commonly decoded by the iterative message passing algorithm also known as the Sum-Product algorithm (SPA) [12]. Each iteration of the algorithm consists of two sequential steps: 1) Row processing or check node update, and 2) column processing or variable node update. We define $V(i) = \{j : H_{ij} = 1\}$ as the set of variable nodes which participate in check equation i. Also $V(i) \backslash j$ denotes all variable nodes in $V(i)$ except node j. The row processing stage updates α messages of all checknodes and sends them to the column processing stage using

$$\alpha_{ij} = \prod_{j' \in V(i) \backslash j} sign(\beta_{ij'}) \times \phi \left(\sum_{j' \in V(i) \backslash j} \phi(|\beta_{ij'}|) \right), \quad (1)$$

where β is the output of column processing stage. In Eq. 1, the magnitude computation of α can be approximated with a minimum function. The algorithm using this approximation is known as the MinSum (MS) algorithm [13].

III. The Multi-Split-Row Decoding Method

To increase parallelism and reduce decoder complexity, the *Multi-Split-Row* method partitions matrix rows into *Spn* multiple blocks (called Split-*Spn*). This requires new circuits to correctly process sign bits among multiple blocks and is especially beneficial for regular high row-weight decoders. A Multi-Split-Row parity check matrix highlighting the row processing operation is shown in Fig. 1. We denote the parity check matrix H divided into Spn partitions columnwise by $H_{Split-Spn}$.

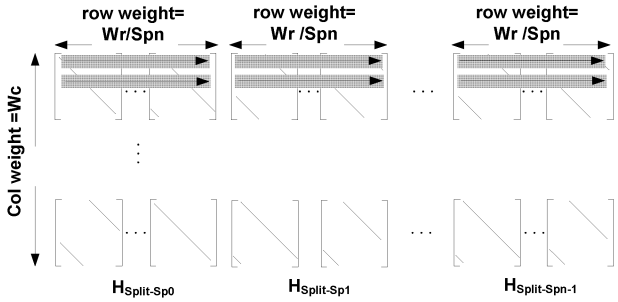

Fig. 1. The parity check matrix of a (W_c, W_r) LDPC code highlighting the row processing operation with *Spn*-way splitting (Multi-Split-Row) method. For simplicity a quasi-cyclic [14], [15] structure is shown.

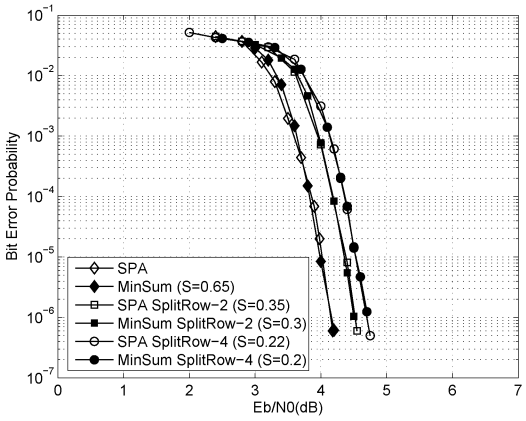

Fig. 2. BER performance of the (6,32) (2048,1723) regular RS-based LDPC code adopted for use in 10GBase-T Ethernet [2], using the Multi-Split-Row method in SPA and MinSum decoders with optimal scaling factors (S).

In each partition of Multi-Split-Row's row operation, the parity (sign) bit update is the same as in the SPA decoder. The magnitude part is updated using the messages in each partition of the parity check matrix. Similar to the standard Split-Row method [7], it can be shown that the magnitude part of the row processor output, α, in the Mult-Split-Row decoder is larger than that in the SPA decoder. Therefore we can improve the error performance of the Multi-Split-Row method by normalizing the α values with a scale factor S less than one. Modifying Eq. 1 using the messages in each partition yields:

$$\begin{aligned} \alpha_{ijSpn} &= S \times \prod_{j' \in V(i) \backslash j} sign(\beta_{ij'}) \\ &\times \phi\left(\sum_{j' \in V_{Split-Spn}(i) \backslash j} \phi(|\beta_{ij'}|) \right) \end{aligned} \quad (2)$$

$V_{Split-Spn}(i) = \{j : H_{ijSplit-Spn} = 1\}$ denotes the set of variable nodes in each partition of the parity check matrix which participates in check equation i.

The Multi-Split-Row method can be used in both SPA and MinSum decoders. Figure 2 shows the error performance loss of SPA Split-Row-2 and Split-Row-4 from SPA is about 0.4 and 0.65 dB at $\mathrm{BER} = 6 \times 10^{-7}$ for the (2048,1723) RS-based LDPC code. Figure 3 shows the error performance of the (5256,4823) code with various numbers of splits using MinSum. The BER degradation increases with the level of splitting. Also, an error floor is observed beginning near $\mathrm{BER} = 2 \times 10^{-6}$ with Split-Row-12 due to too few variable nodes in each partition—which prevents each partition from receiving the majority of the global information and therefore keeps the decoder from converging robustly with high SNR signals.

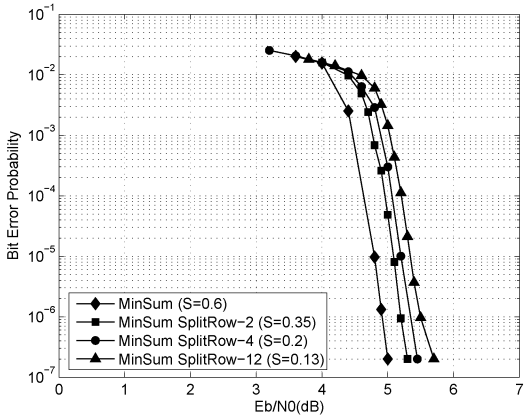

Fig. 3. BER performance of the (6,72) (5256,4823) quasi-cyclic LDPC code using various MinSum decoders with different levels of splitting and optimal scaling factors (S)

A. Implementing Multi-Split-Row Decoders

A block diagram of a decoder using a Spn-way splitting method is shown in Fig. 4. A small number of sign wires are the only wires passed between decoder partitions.

The greatest benefit of Multi-Split-Row comes from the fact that it provides significant reductions in circuit area and wire area (length), which results in reduced overall area and increased throughput. Its reduction in interconnect complexity enables a compounding benefit with an increase in circuit area utilization (that is, a higher % of chip area used for circuits).

1) Sign Wires Implementation: Figure 4 shows the sign wire implementation inside each row processor in the Multi-Split-Row decoder. Local sign signals are generated inside each partition. Final local sign signals are used to compute α messages according to Eq. 2. A block diagram of a Split-Row-*Spn* row processor implemented with the MinSum algorithm is shown in Fig. 6.

2) Area of Sub blocks: The area of individual row processors is largely made up of Muxes, XOR gates, and comparators. Multi-Split-Row provides modest reductions in row processor area and greater reductions in wiring. The total mux area scales down linearly such that the total number of muxes is W_r/Spn where W_r is the row weight and N is the code size. The total number of XORs is $W_r/Spn + 1$. The comparator area scales downward slightly faster than linear with increasing Spn and the total number is $W_r/(2Spn) - 1$.

Column processor area on the other hand is unaffected

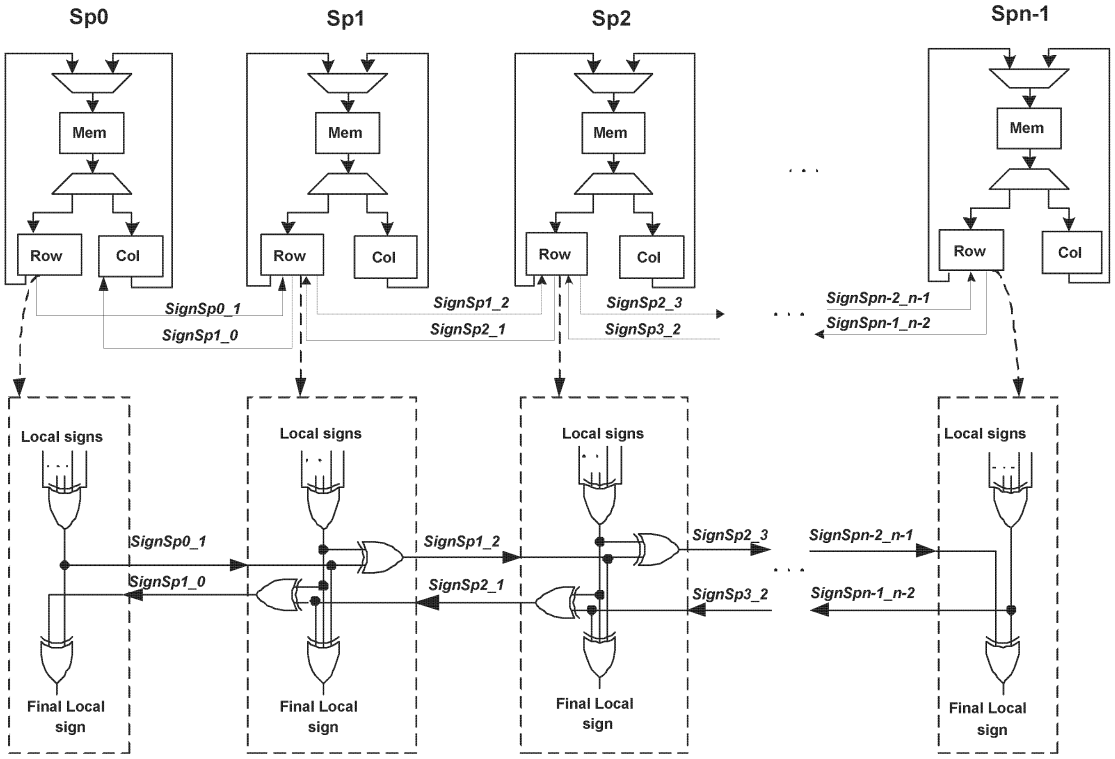

Fig. 4. Multi-Split-Row decoder with *Spn*-way splitting method, with highlighted inter-partition *sign* wires and logic

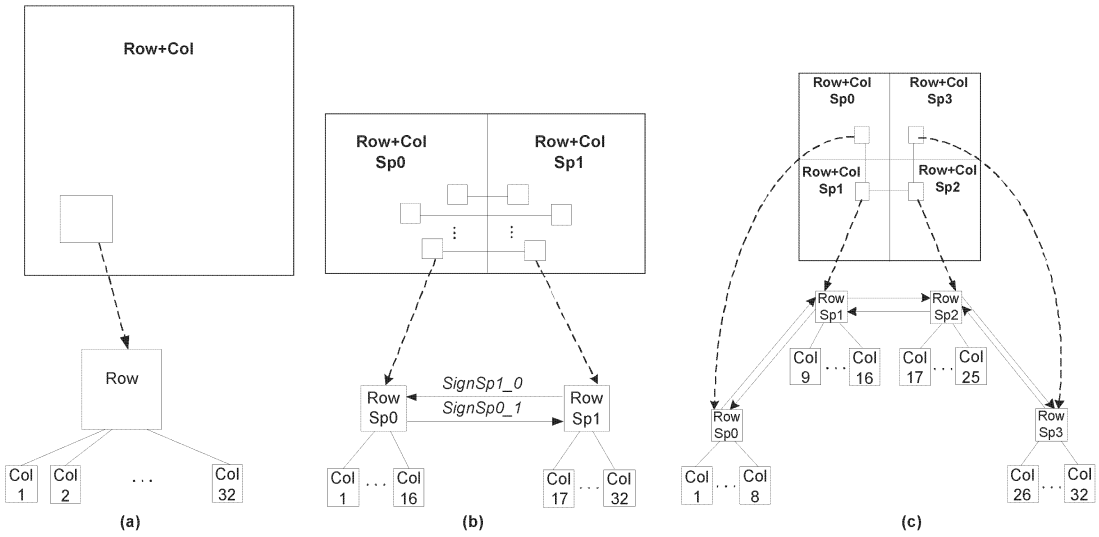

Fig. 5. Mapping an LDPC decoder with (a) standard decoding (b) Split-Row-2 and (c) Split-Row-4 methods for the (6,32) (2048,1723) code

	Maximum area utilization (%)	Area per sub-block (mm^2)	No. of wires per sub-block	Avg. wire length per sub-block (mm)	Total chip area (mm^2)	Worst case speed (MHz)	Decoding throughput (Gbps)
Standard MinSum	25	139.1	122,880	0.32	139.1	10	1.4
Split-Row-2 MinSum	45	37.9	61,440	0.20	75.8	16	2.2
Split-Row-4 MinSum	77	11.0	30,720	0.11	**43.9**	**52**	**7.1**

TABLE I

COMPARISON OF ARCHITECTURES FOR A FULL PARALLEL DECODER IN 0.18 μm CMOS, FOR THE (6,32) (2048,1723) CODE

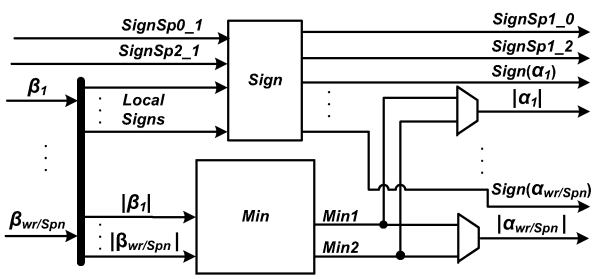

Fig. 6. Block diagram of a row processor in a Split-Row-*Spn* decoder. As an example of a row processor in a general "middle" partition, signal names for partition SP1 are shown (see Fig. 4).

by Multi-Split-Row. The number of column processors per subblock is N/Spn.

3) Wire Area: The total number of wires for a full parallel decoder using the Multi-Split-Row method is:

$$NumOfWires = 2bMW_r + 2(Spn - 1)M, \quad (3)$$

where M is the number of rows and b is the number of bits in each word. The last term in Eq. 3, $2(Spn-1)M$, is the total number of sign wires between row processors for a Split-Row-*Spn* decoder.

Multi-Split-Row is especially effective because the number of sign wires is nearly negligible compared to the total number of wires. For example, for the (2048,1723) LDPC code with $N = 2048, M = 384$ and $b = 5$, the number of sign wires in Split-Row-2 is $2 \times 384 = 768$ which is $768/123,648 = 0.006$ or 0.6% of the total. For Split-Row-4, the number of sign wires is $6 \times 384 = 2304$ which is $2304/125,184 = 0.018$ or less than 2.0% of the total.

B. Decoder Implementation Example

Figure 5 shows block diagrams for decoders using (a) standard, (b) Split-Row-2, and (c) Split-Row-4 decoders for an LDPC code with W_r=32. Using standard decoding, each row processor is connected to 32 column processors. In the Split-Row-2 and Split-Row-4 decoding methods, each row processor is connected to only 16 and 8 column processors respectively. All three decoders are implemented in a 0.18 μm CMOS technology with 6 metal layers. The final layout was generated by a standard cell, place and route (P&R) flow. Table I summarizes the results for the three decoders, and Fig. 7 shows the layout plots for the Multi-Split-Row decoder chips.

IV. Conclusion

The Multi-Split-Row method for regular high row weight LDPC codes is a viable approach for higher throughput, smaller area, and lower power, with a small error performance reduction. The error performance loss by splitting a 5256-bit (6,72) LDPC code into 2, 4 and 12 sub blocks is about 0.3 dB, 0.45 dB and 0.75 dB respectively at a BER of 2×10^{-7}. A (2048,1723) full-parallel decoder using Split-Row-4 is about 3.2 times smaller and delivers 5.2 times higher throughput compared to a standard MinSum decoder, while its BER degradation is 0.65 dB from a standard MinSum decoder.

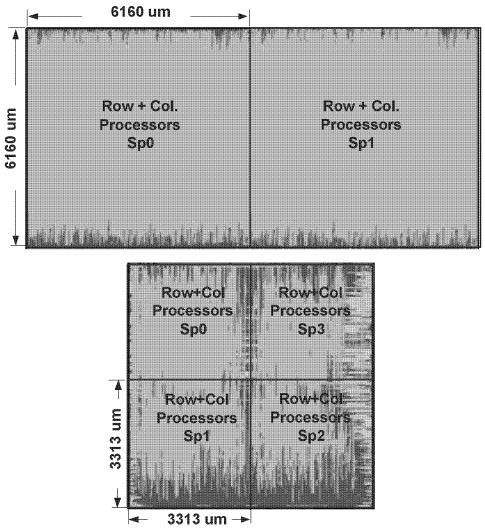

Fig. 7. Final layout of Split-Row-2 (top) and Split-Row-4 (bottom) decoder chips, shown approximately to scale

References

[1] R.G. Gallager, "Low-density parity check codes," *IRE Transaction Info.Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[2] "IEEE P802.3an, 10GBASE-T task force," `http://www.ieee802.org/3/an`.

[3] "T.T.S.I. digital video broadcasting (DVB) second generation framing structure for broadband satellite applications.," `http://www.dvb.org`.

[4] A. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate 1/2 low-density parity-check code decoder," *JSSC*, vol. 37, no. 3, pp. 404–412, Mar. 2002.

[5] A. Darabiha, A.C. Carusone, and F.R. Kschischang, "Multi-Gbit/sec low density parity check decoders with reduced interconnect complexity," in *ISCAS*, 2005, pp. 5194–5197.

[6] E. Kim, N. Jayakumar, P. Bhagwat, and S.P. Khatri, "A high-speed fully-programmable vlsi decoder for regular ldpc codes," in *International Conference on Acoustics, Speech, and Signal Processing*, May 2006, vol. 3, pp. 972–975.

[7] T. Mohsenin and B. Baas, "Split-row: A reduced complexity, high throughput LDPC decoder architecture," in *ICCD*, Oct. 2006.

[8] M. Mansour and N.R. Shanbhag, "A 640-Mb/s 2048-bit programmable LDPC decoder chip," *JSSC*, vol. 41, pp. 684–698, Mar. 2006.

[9] H. Zhong and T. Zhang, "Block-LDPC: A practical LDPC coding system design approach," *IEEE Transaction Circuits and Systems I*, vol. 52, pp. 766–775, Apr. 2005.

[10] Z.Wang, Y. Chen, and K. Parhi, "Scheduling algorithm for partially parallel architecture of ldpc decoder by matrix permutation," in *ISCAS*, May 2005, vol. 6, pp. 5778–5781.

[11] Y. Chen and K. Parhi, "Overlapped message passing for quasi-cyclic low-density parity check codes," *IEEE Transaction Cicuits and Systems I*, vol. 51, pp. 1106–1113, June 2004.

[12] D.J.MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Transaction Info.Theory*, vol. 45, pp. 399–431, Mar. 1999.

[13] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transaction Communications*, vol. 47, pp. 673–680, May 1999.

[14] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-shannon-limit quasi-cyclic low-density parity-check codes," *IEEE Transaction Communications*, vol. 52, pp. 1038–1042, July 2004.

[15] M. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Transaction Info.Theory*, vol. 50, pp. 1788–1793, Aug. 2004.